1. Left factoring & eliminating left recursion

Rewrite the following grammars so they can be parsed by a predicative parser by eliminating left recursion and applying left factoring where necessary

a) S → S + a | b

b) S → S + a | S + b | c

c) S → a b c | a c

d) S → a a | a b | a

e) S → a S c | a S b | b

2. Consider the following grammar.

$$S \to X \mid ay$$
$$X \to xXy \mid Y$$
$$Y \to a$$

(a) Compute the set of SLR(1) items for this grammar and then compute the corresponding DFA. Show the contents of each state (set of items) and label the transitions clearly.

(b) Is this grammar SLR(1)? Briey explain why or why not.

(c) Compute the FIRST sets for x and X, and the FOLLOW set for X.

3. Consider the following CFG, which has the set of terminals T = {stmt, {, }, ;}. This grammar describes the organization of statements in blocks for a fictitious programming language. Blocks can have zero or more statements and other nested blocks, separated by semicolons, where the last semicolon is optional. (P is the start symbol here.)

$$P \to S$$
$$S \to \mathrm{stmt} \mid \{B$$
$$B \to \} \mid S\} \mid S;B$$

(a) Construct a DFA for viable prefixes of this grammar using LR(0) items.

(b) Identify any shift-reduce and reduce-reduce conflicts in this grammar under the SLR(1) rules.

(c) Assuming that an SLR(1) parser resolves shift-reduce conflicts by choosing to shift, show the operation of such a parser on the input string {stmt;}

Best wishes

Dr. Sherin El Gokhy